# BAB II

# LANDASAN TEORI

## 2.1 Anime

Anime adalah suatu bentuk seni yang didefinisikan sebagai sebuah teknik animasi yang dikembangkan oleh Jepang. Asal kata *anime* sampai saat ini masih diperdebatkan. Salah satu sumber yang paling mendekati adalah dari sudut pandang bahasa Inggris. Kata “animasi” diserap ke dalam katakana menjadi animeshon kemudian disingkat menjadi *anime*. Corak seni anime bervariasi dari gaya penggambaran yang berlebihan sampai pendekatan yang realistis. Penggambaran ekspresi wajah yang berlebihan adalah ciri khas dalam teknik penggambaran anime. Anime memiliki lima demografi utama, yaitu "shojo" untung kalangan perempuan, "shonen" untuk kalangan laki-laki, "seinen" untuk kalangan pria muda, “josei” untuk kalangan wanita muda, dan “kodomo” untuk kalangan anak-anak (Bellaonline, 2007).

## 2.2 Pattern Matching in String

Dalam bidang penelitian teknik informatika, *pattern matching* dapat dimengerti sebagai tindakan untuk menemukan suatu *pattern* yang sesuai dengan sebuah urutan *token* yang diberikan (Cambridge, 2002). *Pattern matching* menjadi teknik populer yang digunakan dalam penelitian *chatbot* semenjak *chatbot* generasi pertama, yaitu ELIZA (1996) dengan teknik *keyword pattern matching* sampai *chatbot* generasi terbaru, yaitu ALICE (2007) yang menggunakan teknik *Artificial Intelligence Markup Language* (AIML) *pattern matching*.

### 2.2.1 Artificial Intelligence Markup Language (AIML) Pattern Matching

AIML merupakan bahasa XML-*compliant* yang dapat dengan mudah dipelajari dan memungkinkan pengguna awam untuk mulai mengubah suatu Alicebot atau membuat suatu *chatbot* dari awal dalam beberapa menit. Berdasarkan *chatbot* generasi terbaru, yaitu ALICE (2007) terdapat tiga jenis *pattern* yang diterapkan, yaitu :

**A. Atomic categories**

Kategori ini merupakan *pattern* yang tidak memiliki *wildcard* simbol. *Input text* yang dimasukkan pengguna sama persis dengan *pattern* yang ada.

Contoh:

<pattern> 10 Dollars </pattern>
<template> Wow, that is cheap </temlplate>

Pada kategori ini, jika user melakukan *input* '10 dollars', maka sistem akan menjawab 'Wow, that is cheap'.

**B. Default categories**

Kategori ini merupakan *pattern* yang memiliki *wildcard* simbol. Sebagian *input text* yang dimasukkan pengguna mengandung *pattern* yang ada. Contoh:

<pattern> 10 *</pattern>
<template> It is ten </temlplate>

Pencarian pada kategori ini dilakukan, jika tidak ditemukan kesamaan *pattern* pada *atomic* categories, sehingga sistem akan menjawab 'It is ten'.

## C. Recursive categories

Kategori ini merupakan *pattern* yang memiliki *template* tambahan yang melakukan sebuah aturan rekursif. *Recursive categories* memiliki banyak kegunaan, yaitu:

1) *Symbolic reduction* untuk melakukan reduksi pada tata bahasa yang kompleks. Contoh:

```
<pattern> Do you know what is *</pattern>
<template>
  <srai> What is </srai>
</temlplate>
```

Pada *symbolic reduction,* pertanyaan yang kompleks tersebut direduksi menjad 'What is'.

2) *Divide and Conquer* yang memecah sebuah *input* menjadi dua buah sub bagian, respon dari kedua sub bagian akan digabungkan menjadi satu jawaban. Contoh:

```
<pattern> YES*</pattern>
<template>
  <srai> YES</srai>
</temlplate>
```

Pada *divide anad conquer*, sistem akan menggabungkan jawaban dari *pattern* 'YES' dan 'YES*'.

3) *Synonyms*, kesamaan kata pada *pattern*.

```
<pattern> HALO</pattern>
<template>
  <srai> Hello</srai>
</temlplate>
```

Pada *synonyms*, sistem akan melakukan pencocokkan pada *pattern* 'Hello'.

### 2.2.2 Regular Expression Pattern Matching

Salah satu teknik pencarian lain pada pattern matching adalah menggunakan *regular expression*. *Regular expression* sering digunakan dalam *text retrieval* atau aplikasi penghitungan biologi guna merepresentasikan *search pattern* yang lebih kompleks dari sebuah string atau *extended string*. Menurut Navarro dan Raffinot (2002) *regular expression* memberikan solusi yang sangat kuat dalam mengekpresikan sederet pencarian *pattern*. *Regular Expression Pattern Matching* menggunakan sekumpulan *regular expression* yang disusun menjadi sebuah *pattern*. Bila kalimat masukan cocok dengan salah satu *pattern* yang ada, maka sistem akan melakukan proses sesuai perintah yang ditentukan pada *pattern* tersebut. Gambar 2.1 menampilkan contoh *pattern* menggunakan *regular expression*.

```
$patterns = array(
    'desc' => "/((description)|(summar)|(info)|(story)|(synop))/", // desc
    'type' => "/( ) *((type)|(series)|(movie)|(ova)|(music))/", // type
    'eps' => "/((episode)|(chapter)|(long)|(day))/", /// episode
    'stat' => "/((status))/", /// status
    'date' => "/((date)|(month))/", /// aired
    'season' => "/((season)|(stant)|(aired)|(end)|(period)|(year))/", // season
    'studio' => "/((studio)|(make)|(produce))/", // studio
    'source' => "/(source)|(adapt)/", // source
    'genre' => "/((genre))/", // genre
    'rank' => "/((rank))/", // ranked
    'pop' => "/((popular))/", // popularity
    'url' => "/((link).*((anime)|()))|((url).*((anime)|()))/", // link .... anime
    //
    );
```

Gambar 2.1 Contoh *Regular Expression Pattern*

## 2.3 Natural Language Processing (NLP)

*Natural Language Processing* (NLP) adalah bidang yang dipelajari dalam *computer science, artificial intelligence*, dan *computational linguistics* mengenai interaksi antara komputer dan bahasa natural (natural languages) yang digunakan manusia (Stanford, 2016). Bidang yang membahas mengenai NLP semakin

berkembang dan bertambah banyak seperti *information retrieval* (pencarian informasi) serta *question answering* yang secara umum diadaptasi menjadi sebuah aplikasi berupa *chatbot*. Beberapa teknik utama yang terdapat dalam NLP dalam memproses bahasa natural adalah transformasi, tokenisasi, dan *part of speech tagging* (POS Tagging).

### 2.3.1 Transformasi

Teknik transformasi yang dapat diterapkan untuk mengatasi masalah ini adalah normalisasi, *stop word*, dan *strip punctuation.*

a. Normalisasi dibutuhkan dalam melakukan normalisasi format masukan menjadi huruf kecil. Penerapan normalisasi sangat sederhana yaitu dengan menggunakan fungsi string strtolower(). Normalisasi data dilakukan untuk menjaga konsistensi data di dalam sistem.

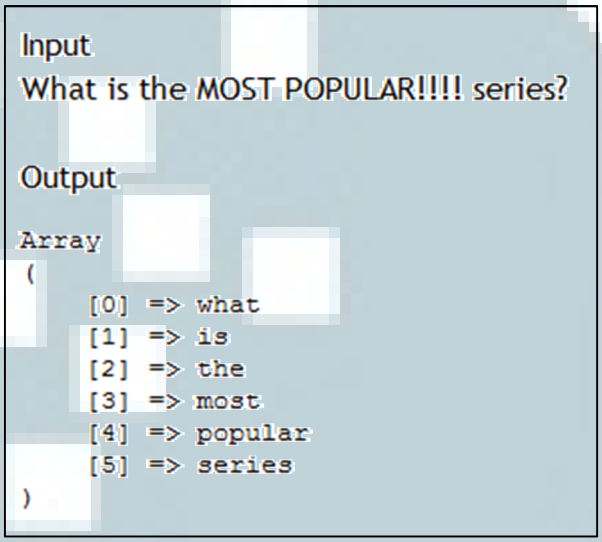
b. *Stop word* dibutuhkan dalam menghilangan kata yang tidak memiliki nilai dan makna terhadap suatu kalimat dapat dihilangkan seperti kata "a, an, in, on". Teknik transformasi ini juga berfungsi dalam menyaring kata-kata kasar atau tidak senonoh yang tidak diperlukan dalam pencarian informasi *anime.*

*c.* *Strip punctuation* dibutuhkan supaya tanda baca tidak menyatu dengan sebuah kata. Hal ini dapat terjadi pada proses tokenisasi.

### 2.3.2 Tokenisasi

Tokenisasi adalah suatu proses pemecahan sebuah teks menjadi bagian yang lebih kecil. Bagian yang lebih kecil ini dikenal dengan istilah *"token"*. Tujuannya adalah untuk memecah suatu kalimat ke dalam sebuah *array* kata (NlpTools, 2016). Tokenisasi sangat dibutuhkan karena sistem tidak hanya menganalisa

keseluruhan kalimat, tetapi menganalisa setiap kata yang membentuk sebuah kalimat. Tahap tokenisasi akan menghasilkan sebuah array kata yang siap diproses oleh fungsi NLP berikutnya seperti *part of speech tagging*. Gambar 2.2 menunjukkan hasil keluaran kalimat setelah transformasi dan tokenisasi diterapkan.




Gambar 2.2 Hasil Tokenisasi Kalimat Natural

### 2.3.3 Part of Speech Tagging

Secara garis besar Universitas Stanford (2016) mengutarakan *Part of Speech Tagging* (POS *Tagging)* sebagai sebuah proses yang bekerja untuk mengidentifikasi tata bahasa pada suatu kalimat. POS *Tagging* digunakan untuk mengelompokkan kata-kata yang terdapat dalam kalimat natural guna memahami kategori kata di dalam suatu kalimat Sebagai contoh berdasarkan tata bahasa natural, kalimat "What is anime?" tersusun atas tiga kata yaitu "what", "is", dan "anime". Pada proses *tagging* ketiga kata tersebut akan diidentifikasi menjadi "*what"* sebagai WH *word*, "is*"* sebagai *verb,* dan "anime" sebagai *noun*. *Noun* merupakan objek dari suatu perbincangan di dalam kalimat sehingga berdasarkan

pengelompokkan di atas sistem dapat mengenali bahwa topik yang terdapat di dalam bahasa natural tersebut adalah "anime". Aplikasi *chatbot* pencarian informasi *anime* (Reikobot) menggunakan Brill Tagger, yaitu sebuah algoritma POS *Tagging* yang menggunakan metode induksi berbasis transformasi untuk memahami tata bahasa natural (Eric Brill, 1995).

Teknik pemberian tag ini menggunakan seperangkat aturan yang telah ditetapkan sebelumnnya dalam bentuk sebuah kumpulan *vocabulary*. Jika kata diketahui, maka kata akan diberikan tag sesuai dengan sebuah kamus *vocabulary* yang telah disiapkan. Jika kata tersebut tidak diketahui, maka secara naif kata tersebut akan diberikan tag *noun* sedangkan jika kata tersebut diketahui, maka kata tersebut akan diberikan tag sesuai *record* pada kamus *vocabulary*. Kemudian, kata tersebut akan memasuki seperangkat *if rules* yang secara bertahap akan mengubah tag awal menjadi tag lain yang lebih akurat bila kata tersebut memenuhi kondisi yang ditentukan pada suatu *rules*. Gambar 2.3 menampilkan contoh pseudocode dari penerapan metode Bril *tagger*.

```
FOR each words as word
    IF the word is set in dictionary THEN
        tag word with dictionary tag
    ELSE
        tag word as noun
    END IF.

    IF word tag as verb and the word before is 'the' THEN
        change word tag to noun
    END IF.

    IF the word before is 'would'
        change word tag to verb
    END IF.

    ..... // any if rules that we want to apply

    IF the word end with 'ly'
        change word tag to adverb
    endif.

    // and so on
END FOR.
```

Gambar 2.3 Contoh Pseudocode Brill Tagger

Tabel 2.1 menampilkan seluruh kategori *tag* dari *part of speech* yang dicakup oleh algoritma Brill Tagger.

Tabel 2.1 *List Part of Speech*

| Number | Tag | Description |
|---|---|---|
| 1. | CC | Coordinating conjunction |
| 2. | CD | Cardinal number |
| 3. | DT | Determiner |
| 4. | EX | Existential *there* |
| 5. | FW | Foreign word |
| 6. | IN | Preposition or subordinating conjunction |
| 7. | JJ | Adjective |
| 8. | JJR | Adjective, comparative |
| 9. | JJS | Adjective, superlative |
| 11. | MD | Modal |
| 12. | NN | Noun, singular or mass |
| 13. | NNS | Noun, plural |
| 14. | NNP | Proper noun, singular |
| 15. | NNPS | Proper noun, plural |
| 16. | PDT | Predeterminer |
| 17. | POS | Possessive ending |
| 18. | PRP | Personal pronoun |
| 19. | PRPS | Possessive pronoun |
| 20. | RB | Adverb |
| 21. | RBR | Adverb, comparative |
| 22. | RBS | Adverb, superlative |
| 23. | RP | Particle |
| 24. | SYM | Symbol |
| 25. | TO | *to* |
| 26. | UH | Interjection |
| 27. | VB | Verb, base form |
| 28. | VBD | Verb, past tense |
| 29. | VBG | Verb, gerund or present participle |
| 30. | VBN | Verb, past participle |

Tabel 2.1 List Part of Speech (Lanjutan)

| Number | Tag | Description |
|---|---|---|
| 31. | VBP | Verb, non-3rd person singular present |
| 32. | VBZ | Verb, 3rd person singular present |
| 33. | WDT | Wh-determiner |
| 34. | WP | Wh-pronoun |
| 35. | WP$ | Possessive wh-pronoun |
| 36. | WRB | Wh-adverb |

## 2.4 Information Retrieval

*Information retrieval* adalah aktivitas yang bertujuan untuk memperoleh sumber informasi yang relevan dengan kebutuhan dari kumpulan sumber informasi. Pencarian dapat didasarkan pada teks atau konten lainnya. Sistem pencarian informasi identik dengan pencarian di dalam koleksi data yang tidak terstruktur atau separuh tidak terstruktur seperti halaman web, dokumen, gambar, video, dan koleksi data lainnya (Sanderson dan Croft, 2012). Perkembangan penyebaran informasi melalui media internet telah mempengaruhi kemajuan teknik pencarian informasi melalui halaman web. Pengambilan data pada halaman web dapat dilakukan dengan menggunakan teknik *web scraping*.

*Web scraping* merupakan sebuah teknik untuk mengekstrak data dan informasi dari sebuah halaman *website*. *Web Scraping* banyak diterapkan karena ruang lingkup informasi yang didapatkan luas, proses ekstraksi data praktis dan informasi yang didapat bersifat dinamis . Proses teknis *web scraping* secara singkat, yaitu pengekstrakan data dimulai dengan cara mengunduh seluruh konten yang ada pada sebuah halaman website. Sesudah itu, dilakukanlah proses *scraping* yang menghilangkan bagian konten yang tidak diperlukan sehingga hanya tersisa konten yang mengandung informasi yang dibutuhkan (*scrape data)*. *Scrape data*

inilah yang selanjutnya akan diproses sesuai dengan keinginan atau kebutuhan pengguna. Berikut merupakan contoh *web scraping* dengan menggunakan PHP dan cURL yang dibuat oleh Jacobward (2013) untuk pengolahan *scraped data*.

1. Pengunduhan Halaman Web

Pengunduhan halaman web merupakan tahapan pertama yang dilakukan di dalam *web scraping*. Proses ini menggunakan cURL yang ditulis dalam bahasa PHP yang berfungsi mengirimkan *request* untuk mengunduh suatu halaman web. Pengunduhan meliputi keseluruhan konten yang ada di dalam halaman web (teks, gambar, dll) yang ditunjukkan oleh Gambar 2.4.

```php
<?php
    // Defining the basic cURL function
    function curl($url) {
        // Initialising cURL
        $ch = curl_init();
        // Setting cURL's URL option with the $url variable passed into the function
        curl_setopt($ch, CURLOPT_URL, $url);
        // Setting cURL's option to return the webpage data
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
        // Executing the cURL request and assigning the returned data to the $data variable
        $data = curl_exec($ch);
        // Closing cURL
        curl_close($ch);
        // Returning the data from the function
        return $data;
    }
?>
```

Gambar 2.4 Contoh cURL function (Jacobward, 2013)

2. Scrape Function

*Scrape function* merupakan fungsi yang dibuat untuk menghilangkan konten yang tidak dibutuhkan dan mengambil data dan informasi yang dianggap berguna. Tujuan utama dari *scrape function* adalah menyediakan sekumpulan data pilihan yang dapat langsung diproses atau digunakan.

```php
<?php
    // Defining the basic scraping function
    function scrape_between($data, $start, $end){
        // Stripping all data from before $start
        $data = stristr($data, $start);
        // Stripping $start
        $data = substr($data, strlen($start));
        // Getting the position of the $end of the data to scrape
        $stop = stripos($data, $end);
        // Stripping all data from after and including the $end of the data to scrape
        $data = substr($data, 0, $stop);
        // Returning the scraped data from the function
        return $data;
    }
?>
```

Gambar 2.6 Contoh scrape function (Jacobward, 2013)

*Scrape function* yang terdapat pada Gambar 2.6 didefinisikan sebagai fungsi scrape_between() yang memiliki tiga buah parameter, yaitu $data, $start, dan $end. Penjelasan secara singkat kinerja setiap kode adalah sebagai berikut.

a. Parameter $data merupakan variabel *string* berisi sumber halaman web yang menjadi target *scraping*, $start merupakan titik dimulainya proses *scrape*, $end merupakan titik selesainya proses *scrape*. $start dan $end mengandung *string* dengan *tag* html yang diisi sesuai dengan *tag* yang menyusun web tersebut, contohnya “ ” dan “</div id=\”sidebar\”>”;

b. stristr() berfungsi untuk melakukan *stripping* atau menghilangkan data yang berada sebelum posisi $start;

c. substr() berfungsi untuk menghilangkan $start dari data yang ada;

d. strpos() berfungsi untuk mendapatkan posisi dari $end, kemudian substr() menghilangkan $end dan data setelahnya sehingga hanya tersisa data yang dibutuhkan dalam sebuah variabel $data.

## 2.5 Chatbot

*Chatbot* adalah program komputer yang dirancang untuk menstimulasi percakapan intelektual dengan satu atau lebih manusia baik secara audio maupun teks (Mauldin, 1994). Pengembangan teknologi *chatbot* didasari oleh pernyataan yang dikeluarkan Zadronny (2000) yang menyatakan bahwa cara terbaik untuk memfasilitasi interaksi manusia dan komputer adalah dengan memberikan kenyamanan bagi pengguna untuk dapat mengekspresikan ketertarikan, keinginan, atau pertanyaan natural, dengan cara berbicara, mengetik, dan menerima respon secara interaktif.

*Chatbot* berinteraksi dengan menerjemahkan bahasa natural yang diajukan pengguna. Bahasa natural adalah bahasa yang secara umum digunakan oleh manusia dalam berkomunikasi satu sama lain (Oxford, 2016). Salah satu cabang ilmu *Artificial Intelligence* (AI) yang berfokus pada pengolahan bahasa natural adalah *Natural Language Processing* (NLP). Beberapa teknik NLP dalam pencarian informasi berbasis teks, diantaranya:

### A. Chatbot sebagai Aplikasi Sistem Pakar

*Chatbot* pertama yang diciptakan adalah ELIZA, sebuah sistem untuk meniru psikoterapis di pengobatan klinis yang diciptakan oleh Joseph Weizenbaum (Weizenbaum, 1966). Tujuan pertama dibentuknya *chatbot* adalah sebagai sebuah sistem yang dapat berkomunikasi dengan manusia menggunakan bahasa natural dan menghibur penggunanya. Teknik yang digunakan ELIZA merupakan penerapan dari *keyword matching* sederhana.*Chatbot* dibangun dengan inspirasi untuk membodohi pengguna bahwa mereka sedang berkomunikasi dengan manusia lain (Mauldin, 1994).

```
Human:  Men are all alike.
ELIZA: In what way?
Human:  They're always bugging us about something.
ELIZA: Can you think of a special example?
Human:  Well, my boyfriend made me come here.
ELIZA: Your boyfriend made you come here?
Human:  He says I'm depressed much of the time.
ELIZA: I'm sorry to hear that you are depressed much of the time.
Human:  It's true.  I'm unhappy.
ELIZA: Do you think coming here will help you not to be unhappy?
Human:  I need some help, that much seems certain.
```

Gambar 2.7 Sampel Pembicaraan dengan ELIZA
(Jurafsky dan Martin, 2000)

## B. Chatbot sebagai Media Pencarian Informasi

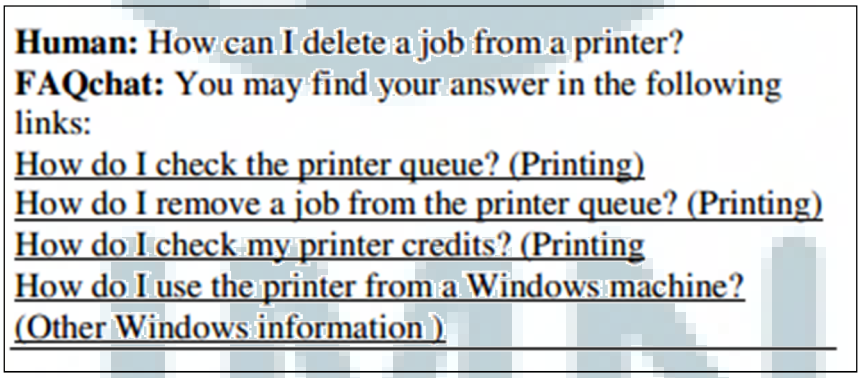
Sebuah *chatbot* pencarian informasi yang telah berhasil dibangun adalah FAQchat (Shawar dan Atwell, 2005) yang digunakan sebagai sebuah aplikasi *question answering* seputar Frequently Ask Question (FAQ) di sebuah *School of Computing* (SoC), University of Leeds. FAQchat adalah sebuah *chatbot* pencarian informasi yang dikembangkan dengan menggunakan AIML (*Artificial Intelligence Markup Language*) *pattern matching* yang mengambil informasi dari *databae* FAQ (*Frequently Ask Question*). FAQchat memperoleh informasi menggunakan kata yang signifikan sebagai kata kunci, kemudian mencoba mencari *pattern* terpanjang yang sesuai tanpa menggunakan *linguistic tools* atau menganalisa makna dari *query* pengguna. FAQchat tidak membutuhkan modul pengetahuan *linguistic* dan menerapkan prinsip *language independent* . Proses kerja FAQchat adalah sebagai berikut (Shawar, 2005):

1) Semua pertanyaan dan jawaban diekstrak dari database setelah melakukan process *filtering* untuk menghilangkan *tag* yang tidak dibutuhkan.

2) *Database* FAQ tersusun atas pertanyaan dan jawaban. Dengan menggunakan pola tersebut, sebuah daftar *link* dibangun, yang berisi *link* dari FAQ ke halaman web yang berisi jawaban.

3) Disusunlah sebuah kamus yang berisikan semua kata dalam pertanyaan dengan frekuensi kejadian. Kemudian, kata pertama atau kedua yang paling signifikan diambil dari setiap pertanyaan yang telah diajukan.

4) AIML *patern matching rules,* atau dikenal sebagai "kategori" dibuat. Terdapat dua tipe *pattern matching* pada FAQchat, yaitu seluruh pertanyaan, atau hanya kata pertama atau kedua yang paling signifikan yang mengalami kecocokan dengan data pertanyaan pada *database* FAQ. Terdapat dua kemungkinan respon yang diberikan FAQchat, yaitu sebuah jawaban jika terdapat satu pertanyaan yang cocok atau beberapa jawaban jika kata pertama atau kedua yang paling signifikan ditemukan dalam beberapa pertanyaan.


**Human:** How can I delete a job from a printer?
**FAQchat:** You may find your answer in the following links:
How do I check the printer queue? (Printing)
How do I remove a job from the printer queue? (Printing)
How do I check my printer credits? (Printing
How do I use the printer from a Windows machine?
(Other Windows information )

Gambar 2.8 Sampel Jawaban FAQchat (Shawar dkk., 2005)

FAQchat bukanlah sebuah mesin pencari, melainkan sebuah media yang dapat mengakses halaman web dan memberikan jawaban dari *database* FAQ. FAQchat diciptakan tidak dengan tujuan mendemonstrasikan bahwa FAQchat dapat menjadi alternatif sebagai media pencarian informasi (Shawar dkk., 2005).

## 2.6 Presisi, Recall dan Harmonic Mean

Menurut Manning dkk. (2008) dalam bukunya yang berjudul *Introduction to Information Retrieval* unsur utama yang diukur dalam sistem pencarian informasi adalah presisi dan *recall*. Presisi merupakan seberapa banyak jumlah dokumen relevan yang didapatkan dari seluruh dokumen yang berhasil diambil. *Recall* mendefnisikan seberapa banyak sistem mengembalikan dokumen yang relevan dari seluruh dokumen relevan yang tersedia.

Tabel 2.2 Tabel Kontingensi

| | Relevant | Non Relevant |
|---|---|---|
| Retrieved | true positive (tp) | false positive (fp) |
| Non Retrieved | false negative (fn) | true negative (tn) |

Berdasarkan Tabel Kontingensi (Tabel 2.2) gagasan presisi dan *recall* dapat ditafsirkan menjadi sebuah persamaan berikut.

$$\text{Presisi} = \frac{\#(relevant\ items\ retrieved)}{\#(retrieved\ items)} = P(\frac{tp}{tp+fp}) \quad \ldots(2.1)$$

$$Recall = \frac{\#(relevant\ items\ retrieved)}{\#(relevant\ items)} = P(\frac{tp}{tp+fn}) \quad \ldots(2.2)$$

Alternatif lainnya yang mungkin dipikirkan pembaca adalah menilai sebuah sistem pencarian informasi dengan menggunakan akurasi, yaitu penghitungan berdasarkan klasifikasi sejumlah dokumen yang tepat.

$$Accuracy = \frac{tp+tn}{tp+fp+fn+tn} \quad \ldots(2.3)$$

Hal ini tampak masuk akal karena penghitungan akurasi menggunakan seluruh unsur relevan dan non relevan yang ada di dalam Tabel Kontingensi. Namun, terdapat alasan yang baik mengapa akurasi bukanlah langkah yang tepat dalam menangani masalah pencarian informasi. Manning dkk. (2008) mengungkapkan bahwa hampir dalam semua kondisi, data yang tersedia sangat

ambigu, normalnya lebih dari 99.9% dokumen termasuk ke dalam kategori *non-relevant*. Peningkatan akurasi sebuah sistem dapat dilakukan secara mudah hanya dengan menganggap semua dokumen yang diambil sebagai dokumen *non-relevant.* Akan tetapi, memberi semua dokumen dengan label *non-relevant* sungguh tidak memuaskan bagi pengguna pencarian informasi. Pengguna ingin selalu melihat dokumen-dokumen dan memiliki toleransi tertentu untuk melihat dokumen *false positive* dengan imbalan bahwa mereka mendapatkan beberapa informasi yang berguna.

Keuntungan yang didapat dari dua buah unsur evaluasi (presisi dan *recall*) dalam pencarian informasi adalah satu dengan yang lain saling melengkapi dalam kondisi-kondisi tertentu. Oleh karena presisi dan *recall* merupakan unsur yang saling berlawanan (*trade*-off). Dalam mengevaluasi toleransi *trade-off* tersebut digunakanlah penghitungan F-Measure yang merupakan *weighted harmonic mean* dari presisi dan *recall*. Nilai minimum *harmonic mean* yang harus dicapai adalah sebesar 70% dikarenakan pada titik tersebut nilai dari presisi dan recall mencapai keseimbangan *trade-off* (Rijsbergen, 1979).

$$F - Measure = \frac{2PR}{P + R} \quad \ldots (2.4)$$

## 2.7 HMSAM (Hedonic Motivation System Adoption Model)

HMSAM adalah sebuah model untuk mengukur sebuah sistem yang mengadaptasi motivasi hedonis. Ada lima faktor yang menjadi fokus pengukuran pada HMSAM, yaitu *perceived usefulness, perceived ease of use, curiosity, control,* dan *joy* yang akan mempengaruhi *behavioral intentional to use* dan *immersion* suatu aplikasi (Lowry dkk., 2013).

a. *Perceived usefulness*, mengukur peningkatan kinerja ketika menggunakan suatu sistem.

b. *Perceived ease of use,* mengukur kemudahan penggunaan suatu sistem.

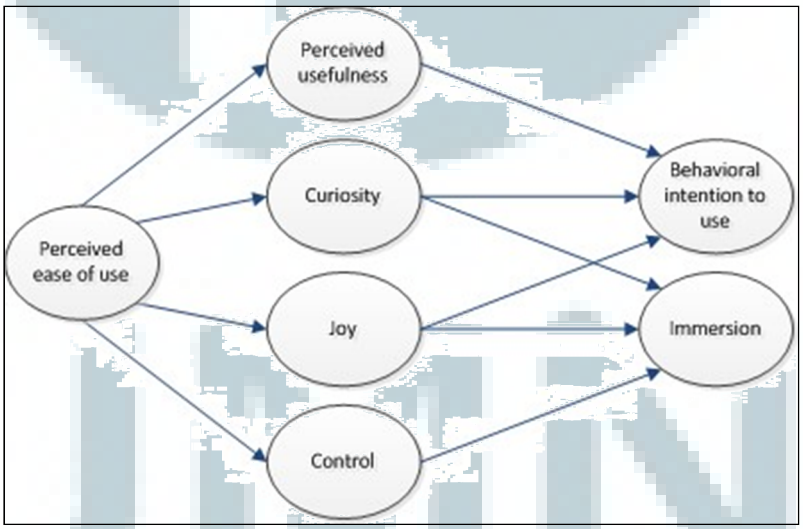
c. *Curiosity,* sejauh mana suatu sistem dapat meningkatkan rasa ingin tahu dalam aspek kognitif.

d. *Control,* persepsi pengguna bahwa dirinya yang diajak berinteraksi oleh sistem.

e. *Joy,* aspek kesenangan yang didapatkan dari interaksi dengan sistem.

f. *Behavioral intentional to use,* keinginan pengguna untuk menggunakan aplikasi.

g. *Focussed Immersion,* total keterlibatan yang dilakukan antara pengguna dengan aplikasi yang mengukur seberapa dalam pengguna terfokus dalam menggunakan sistem.




Gambar 2.9 Overview of HMSAM (Lowry dkk., 2013)

Aspek utama yang dinilai di dalam HMSAM adalah aspek *Behavioral Intention to Use* dan *Immersion*. Penghitungan kedua buah aspek tersebut disajikan dalam rumus berikut.

$$X = \frac{(PEOU) + (PU) + (Curiosity) + (Joy)}{nAspek} \quad \ldots (2.5)$$

$$X = \frac{(PU) + (Curiosity) + (Joy) + (Control)}{nAspek} \quad \ldots (2.6)$$

## 2.8 Skala Likert

Menurut Sugiyono (2014) skala Likert merupakan sebuah metode untuk mengukur data kualitatif menjadi kuantiatif. Skala ini menggunakan model *rating* dalam pengukurannya. Model *rating* menyediakan jawaban yang bersifat kuantitatif. Dengan demikian, jawaban dari pertanyaan yang bersifat kualitatif dapat disajikan dalam sebuah data kuantitatif. Skala Likert menggunakan nilai positif dan negatif sebagai batas pengukurannya. Pemodelan *rating* dalam skala Likert secara umum menggunakan lima buah poin penilaian, dimulai dari nilai tidak setuju sampai setuju dengan nilai tengahnya berupa nilai netral. Penghitungan skor pada Likert (1932) dibagi menjadi lima kriteria yang ditunjukkan pada Tabel 4.3 dengan 'X' sebagai nilai.

Tabel 4.3 Kriteria Likert Scale

| Kategori | Kriteria | Syarat |
|---|---|---|
| SA | Strongly Agree | $X \geq 80\%$ |
| A | Agree | $60\% \leq= X < 80\%$ |
| N | Neither | $40\% \leq X < 60\%$ |
| D | Disagree | $20\% \leq= X < 40\%$ |
| SD | Strongly Disagree | $0\% \leq= X < 20\%$ |

Berdasarkan Tabel Kriteria Likert Scale, rumus yang digunakan untuk menghitung skor adalah sebagai berikut.

$$X = \frac{(SA * nSA) + (A * nA) + (N * nN) + (D * nD) + (SD * nSD)}{\text{Jumlah Pertanyaan} * \text{Jumlah Sampel}} \quad \ldots (2.7)$$